L Number	Hits	Search Text	DB	Time stamp
7	23	"reference array" and (direct adj0 memory adj0 access)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:05
8	21	"moving garbage" and cache	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:12
9	23	(without near4 cache) same "garbage collection"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:29
10	30	without adj1 stor$3 adj2 cache	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:56
11	525	live with cache	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:44
12	2713	garbage same object	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:45
13	41	(live with cache) and (garbage same object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 17:45
14	3	(live with cache) with garbage	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 18:00
16	1	((live with cache) with location) with new	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 18:01
15	9	(live with cache) with location	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/09/24 18:01

Terms used **bypassing cache**

Found **10** of **169 searched out of 169.**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 10 of 10

Relevance scale

1 Cache performance of fast-allocating programs

Marcelo J. R. Gonçalves, Andrew W. Appel

October 1995 **Proceedings of the seventh international conference on Functional programming languages and computer architecture**

Full text available: pdf(1.47 MB) Additional Information: full citation, references, citings, index terms

2 Some issues and strategies in heap management and memory hierarchies

Paul R. Wilson

January 1991 **ACM SIGPLAN Notices**, Volume 26 Issue 3

Full text available: pdf(802.62 KB) Additional Information: full citation, citings, index terms

3 Reducing garbage collector cache misses

Hans-J. Boehm

October 2000 **ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management**, Volume 36 Issue 1

Full text available: pdf(774.19 KB) Additional Information: full citation, abstract, citings, index terms

Cache misses are currently a major factor in the cost of garbage collection, and we expect them to dominate in the future. Traditional garbage collection algorithms exhibit relatively little temporal locality; each live object in the heap is likely to be touched exactly once during each garbage collection. We measure two techniques for dealing with this issue: prefetch-on-grey, and lazy sweeping. The first of these is new in this context. Lazy sweeping has been in common use for a decade. It ...

4 Concurrent compacting garbage collection of a persistent heap

James O'Toole, Scott Nettles, David Gifford

December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles**, Volume 27 Issue 5

Full text available: pdf(1.50 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a replicating garbage collector for a persistent heap. The garbage collector cooperates with a transaction manager to provide safe and efficient transactional storage management. Clients read and write the heap in primary memory and can commit or abort their write operations. When write operations are committed they are preserved in stable storage and survive system failures. Clients can freely access the heap during garbage collection because the collector concurrently builds a comp ...

5 Dynamic class loading in the Java virtual machine

Sheng Liang, Gilad Bracha

October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 33 Issue 10

Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, citings, index terms

Class loaders are a powerful mechanism for dynamically loading software components on the Java platform. They are unusual in supporting all of the following features: *laziness, type-safe linkage, user-defined extensibility*, and *multiple communicating namespaces*.We present the notion of class loaders and demonstrate some of their interesting uses. In addition, we discuss how to maintain type safety in the presence of user-defined dynamic class loading.

6 Reconsidering custom memory allocation

Emery D. Berger, Benjamin G. Zorn, Kathryn S. McKinley

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 37 Issue 11

Full text available: pdf(344.86 KB) Additional Information: full citation, abstract, references, citings, index terms

Programmers hoping to achieve performance improvements often use custom memory allocators. This in-depth study examines eight applications that use custom allocators. Surprisingly, for six of these applications, a state-of-the-art general-purpose allocator (the Lea allocator) performs as well as or better than the custom allocators. The two exceptions use regions, which deliver higher performance (improvements of up to 44%). Regions also reduce programmer burden and eliminate a source of memory ...

7 Session 5: P2P and streaming: A hierarchical characterization of a live streaming media workload

Eveline Veloso, Virgílio Almeida, Wagner Meira, Azer Bestavros, Shudong Jin

November 2002 **Proceedings of the second ACM SIGCOMM Workshop on Internet measurment**

Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, index terms

We present what we believe to be the first thorough characterization of *live* streaming media content delivered over the Internet. Our characterization of over 3.5 million requests spanning a 28-day period is done at three increasingly granular levels, corresponding to clients, sessions, and transfers. Our findings support two important conclusions. First, we show that the nature of interactions between users and objects is fundamentally different for live versus stored objects. Access to ...

8 Interprocedural compatibility analysis for static object preallocation

Ovidiu Gheorghioiu, Alexandru Salcianu, Martin Rinard

January 2003 **ACM SIGPLAN Notices , Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 38 Issue 1

Full text available: pdf(277.65 KB) Additional Information: full citation, abstract, references, citings, index terms

We present an interprocedural and compositional algorithm for finding pairs of *compatible* allocation sites, which have the property that no object allocated at one site is live at the same time as any object allocated at the other site. If an allocation site is compatible with itself, it is said to be *unitary*: at most one object allocated at that site is live at any given point in the, execution of the program. We use the results of the analysis to statically preallocate memory spa ...

Keywords: interprocedural analysis, memory preallocation, static analysis

9 Reverse interpretation + mutation analysis = automatic retargeting

Christian S. Collberg

May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation**, Volume 32 Issue 5

Full text available: pdf(1.92 MB) Additional Information: full citation, abstract, references, citings, index terms

There are three popular methods for constructing highly retargetable compilers: (1) the compiler emits abstract machine code which is interpreted at run-time, (2) the compiler emits C code which is subsequently compiled to machine code by the native C compiler, or (3) the compiler's code-generator is generated by a back-end generator from a formal machine description produced by the compiler writer.These methods incur high costs at run-time, compile-time, or compiler-construction time, respectiv ...

10 Automatic derivation of compiler machine descriptions

Christian S. Collberg

July 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 4

Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, index terms, review

We describe a method designed to significantly reduce the effort required to retarget a compiler to a new architecture, while at the same time producing fast and effective compilers. The basic idea is to use the native C compiler at compiler construction time to discover architectural features of the new architecture. From this information a formal machine description is produced. Given this machine description, a native code-generator can be generated by a back-end generator such as BEG or burg ...

Keywords: Back-end generators, compiler configuration scripts, retargeting

Results 1 - 10 of 10

Your search matched **1** of **1074479** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or entering a new one in the text box.

live object and cache Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

1 **Tracking object life cycle for leakage energy optimization**
Chen, G.; Vijaykrishnan, N.; Kandemir, M.; Irwin, M.J.; Wolczko, M.;
Hardware/Software Codesign and System Synthesis, 2003. First IEEE/ACM/IFIP International Conference on , 1-3 Oct. 2003
Pages:213 - 218

[Abstract] [PDF Full-Text (539 KB)] **IEEE CNF**